**NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY**

Distributed Computing

(CACSC15))

LAB FILE

**Name: SNEHA GUPTA**

**Roll No.: 2021UCA1859**

Branch: COMPUTER SCIENCE WITH ARTIFICIAL INTELLIGENCE

Section: 1

INDEX

1-Program to implement non token based algorithm for Mutual Exclusion

2-Program to implement Lamport’s Logical Clock

3-Program to implement edge chasing+ distributed deadlock detection algorithm.

4-Program to implement locking algorithm.

5-Program to implement Remote Method Invocation.

6-Program to implement Remote Procedure Call.

7-Program to implement Chat Server.

8-Program to implement termination detection

9-To implement CORBA mechanism by using C++ program at one end and Java Program on the other.

**Q1) Program to implement non token based algorithm for Mutual Exclusion**

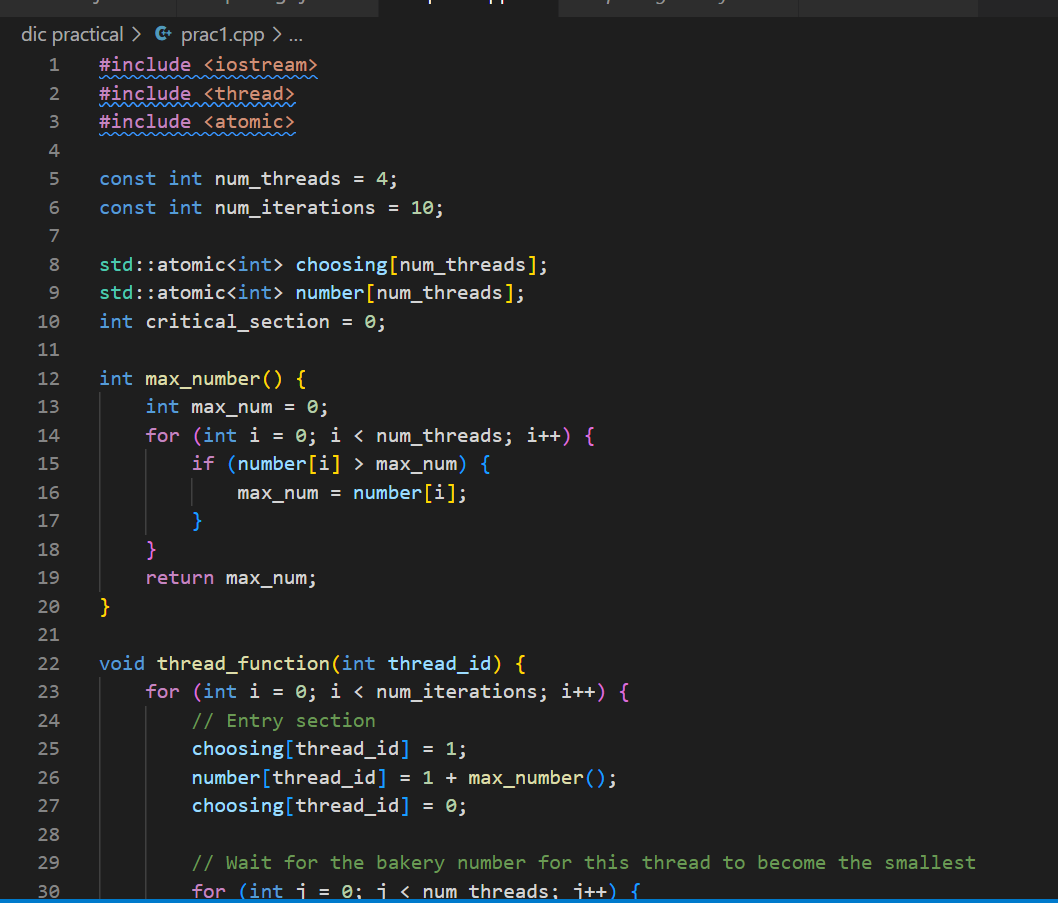
***A non-token based algorithm*** for mutual exclusion aims to ensure that only one process can access a critical section at a time without using tokens. One such algorithm is the ***Lamport's Bakery Algorithm.***

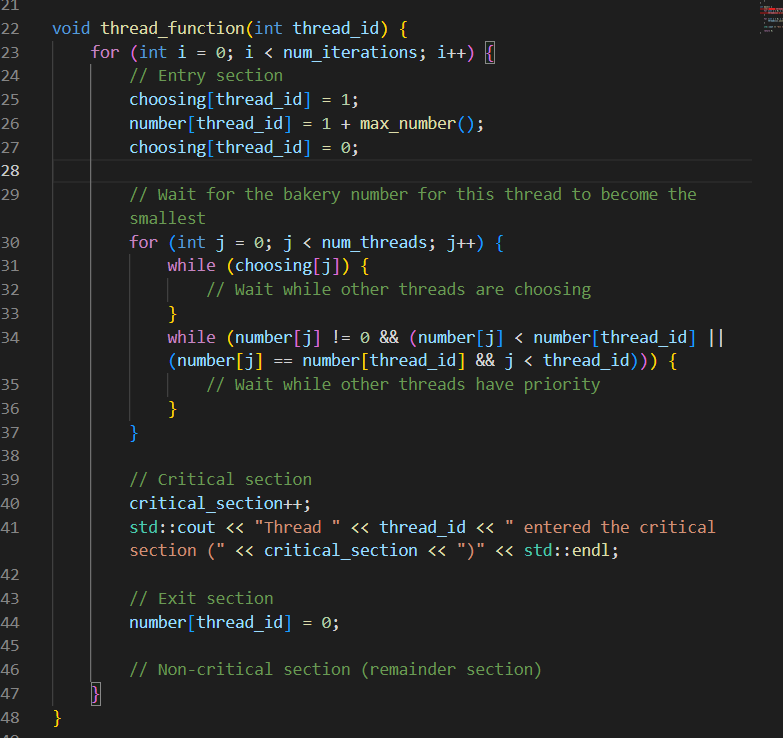
A non-token based algorithm, in the context of mutual exclusion and distributed systems, is a mechanism for coordinating and controlling access to shared resources without the use of explicit tokens. Token-based algorithms rely on the passing of tokens, physical or virtual, among processes or threads to determine which one has the right to access a critical section or a shared resource. Non-token based algorithms, on the other hand, achieve mutual exclusion through other means, often without the need for a token to be explicitly passed.

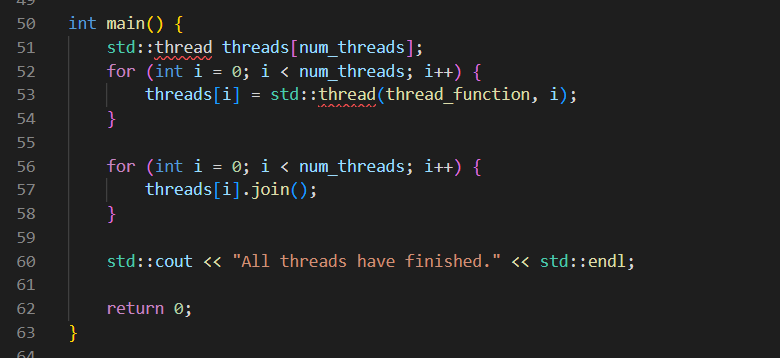
***A non-token based algorithm for mutual exclusion*** is a mechanism or approach used to ensure that only one process can access a critical section or shared resource at a time without the use of explicit tokens. In other words, it achieves mutual exclusion without relying on a physical or logical token that is passed between processes to indicate permission for access. Non-token based algorithms use different methods and techniques to coordinate and control access to shared resources. One of the most well-known non-token based algorithms is Lamport's Bakery Algorithm, which I mentioned earlier.

***Lamport's Bakery Algorithm***, named after its creator, Leslie Lamport, is a classical and simple algorithm for achieving mutual exclusion among multiple processes or threads. It ensures that only one process can access a critical section at a time without the use of tokens. This algorithm is often used in concurrent computing and distributed systems to prevent race conditions and guarantee exclusive access to shared resources.

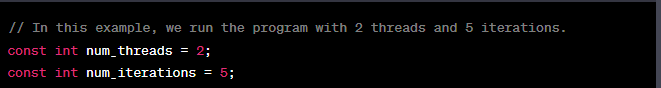
**C++ CODE:**



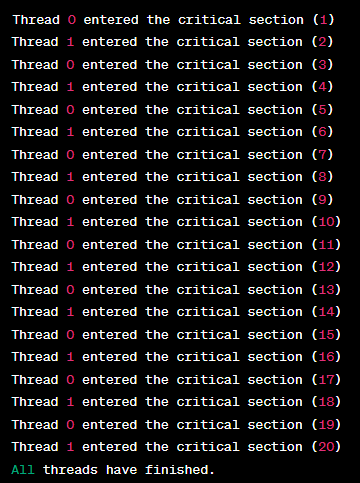




**INPUT:**



**OUTPUT:**



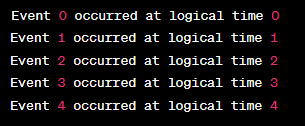
**2-Program to implement Lamport’s Logical Clock**

***Lamport's Logical Clock***, introduced by computer scientist Leslie Lamport in 1978, is a concept used in distributed systems to establish a partial order of events in a distributed system. It is not a physical clock that keeps track of time but rather a mechanism for maintaining a logical order of events based on timestamps. The primary purpose of Lamport's Logical Clock is to provide a way to reason about the relative order of events in a distributed system, even when there is no global clock or synchronized time.

**C++ CODE**



OUTPUT:



**3-Program to implement edge chasing distributed deadlock detection algorithm.**

***The Edge Chasing Distributed Deadlock Detection Algorithm*** is a distributed algorithm used to detect and resolve deadlocks in a distributed computing environment. It is specifically designed to work in distributed systems where processes or nodes are interconnected and communicate with each other. Deadlocks can occur when multiple processes are waiting for resources held by other processes, resulting in a situation where no process can make progress.

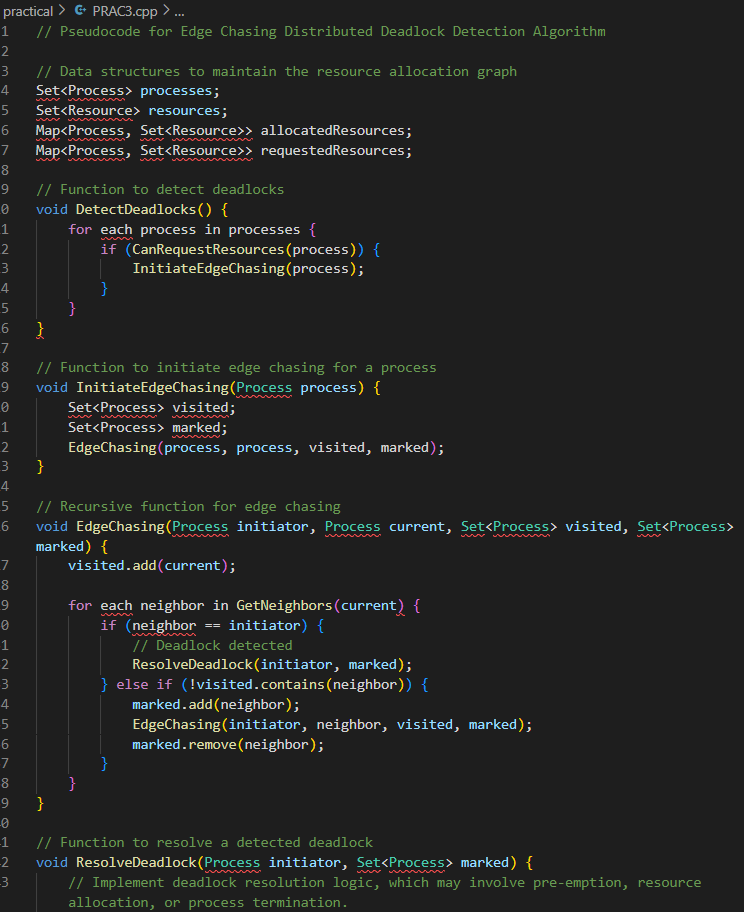
Here are the ***key characteristics and principles*** of the Edge Chasing Distributed Deadlock Detection Algorithm:

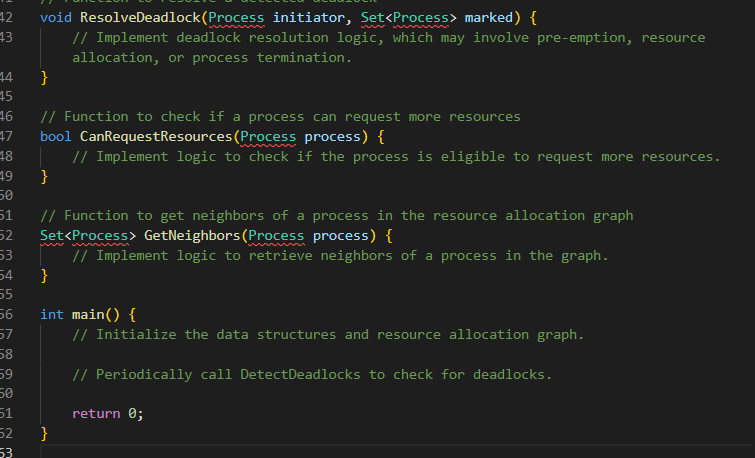
***Resource Allocation Graph:*** In a distributed system, resources are allocated to processes, and a resource allocation graph is maintained. Each process and resource is represented as nodes in the graph, and edges denote the allocation and request of resources.

***Edge Chasing***: The algorithm is named "Edge Chasing" because it works by chasing edges in the resource allocation graph. It starts at a node (typically a process) and traverses the edges to detect cycles in the graph.

***Cycle Detection***: The primary objective of the algorithm is to detect cycles in the resource allocation graph. A cycle in the graph indicates the presence of a potential deadlock situation.

**C++ CODE:**

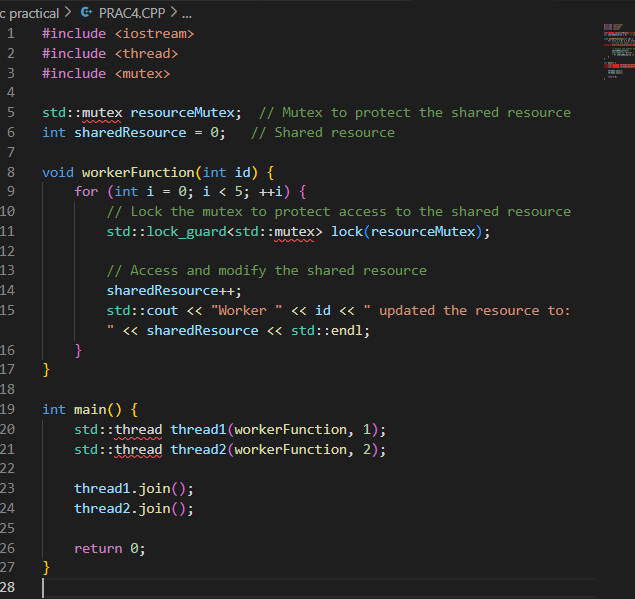




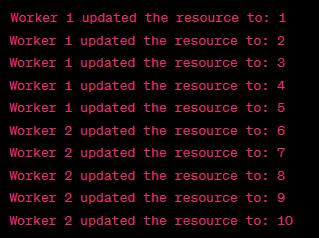
**4-Program to implement locking algorithm.**

***A locking algorithm***, in the context of concurrent programming and operating systems, is a mechanism used to coordinate access to shared resources in a way that prevents conflicts, race conditions, and data corruption. Locking algorithms are crucial for ensuring mutual exclusion, which means that only one thread or process can access a critical section of code or a shared resource at a time. Locks are used to enforce mutual exclusion by allowing one thread to enter a critical section while blocking others until the lock is released.

**C++ CODE:**



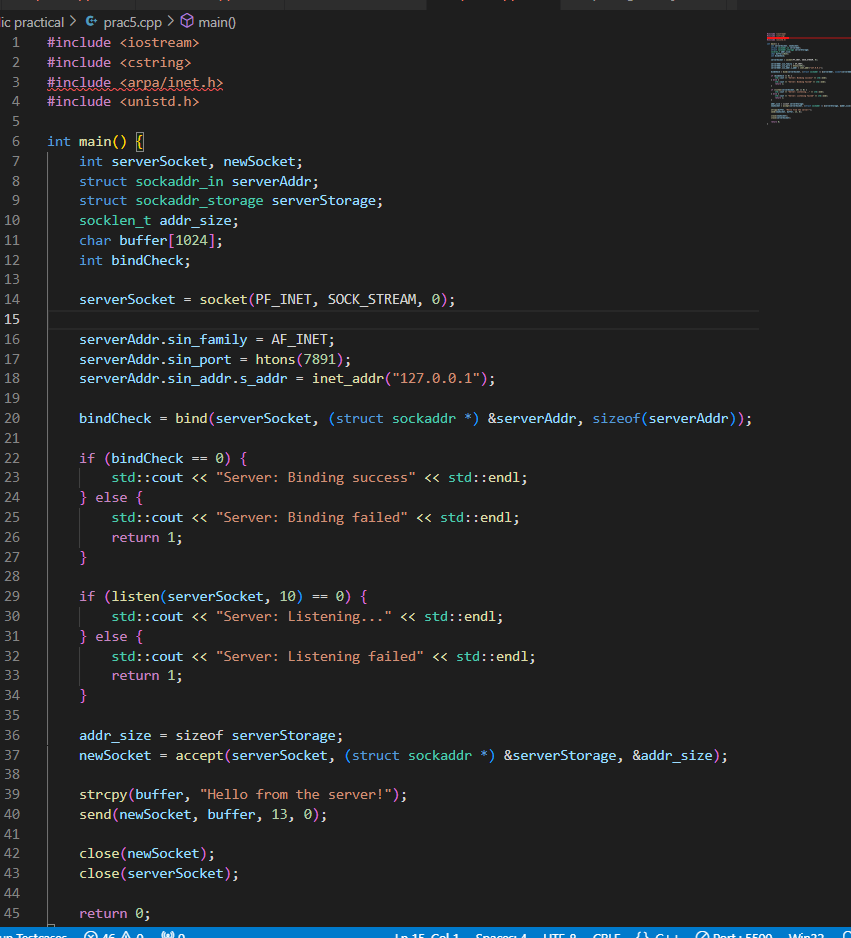
**OUTPUT:**



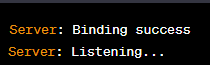
**5-Program to implement Remote Method Invocation.**

***Remote Method Invocation (RMI)*** is a Java-based mechanism that allows one Java virtual machine (JVM) to invoke methods on objects located in another JVM, which can be running on a different physical or networked machine. RMI enables distributed computing by providing a way for objects in one JVM to interact with objects in another JVM as if they were local, while abstracting away the complexities of network communication.

**C++ code:**



**OUTPUT:**

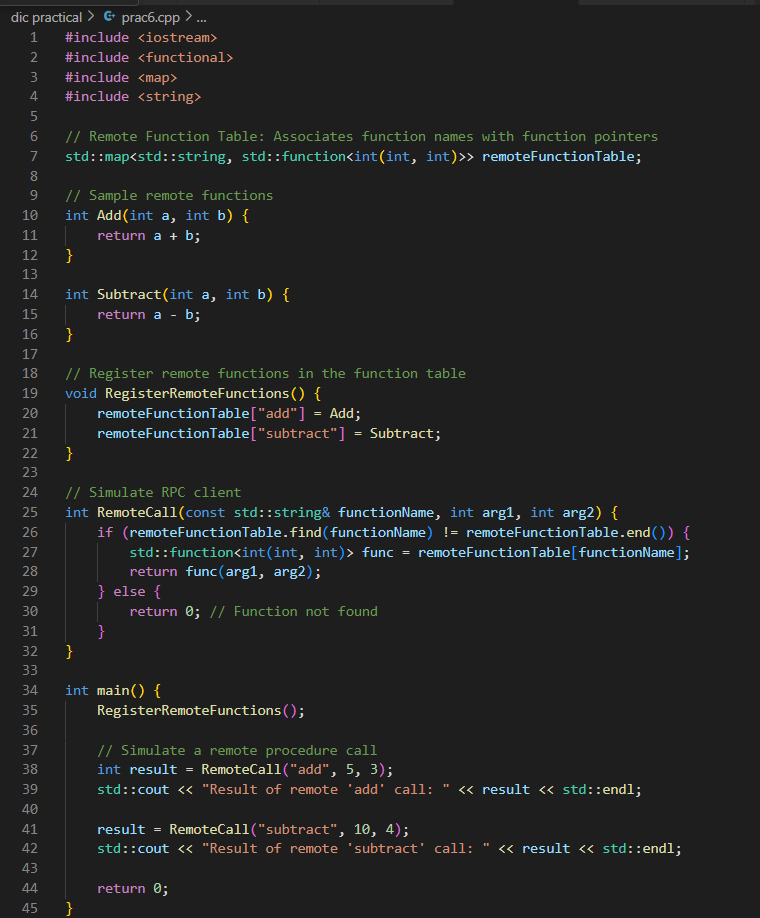


**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**6-Program to implement Remote Procedure Call.**

A Remote Procedure Call (RPC) is a protocol that allows a program to execute a procedure (subroutine or method) on a remote server or another address space, as if it were a local procedure call, without the programmer explicitly coding the details of remote communication. RPC is a high-level, language-independent way to make distributed computing easy and efficient.

**C++ CODE**

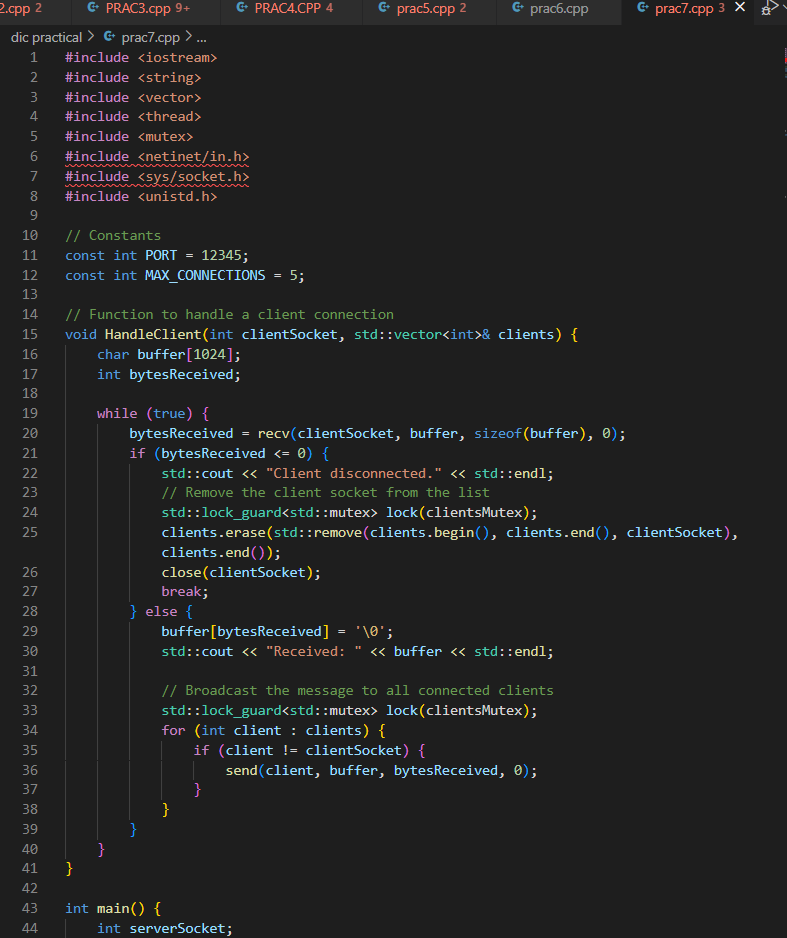


**OUTPUT:**



**7-Program to implement Chat Server.**

Implementing a chat server is a more complex task that typically involves multiple clients connecting to a central server, which facilitates communication between clients. Below is a simplified C++ program for a basic chat server and client using sockets. This example allows multiple clients to connect to the server and exchange messages. It does not include features like user authentication, error handling, or advanced functionality, but it provides a starting point for a chat server.

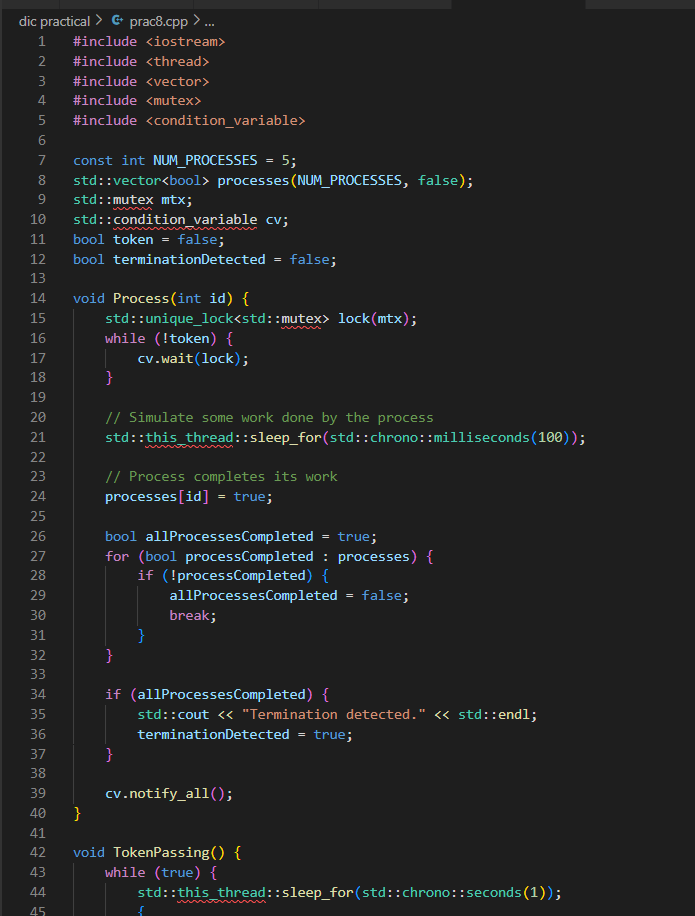


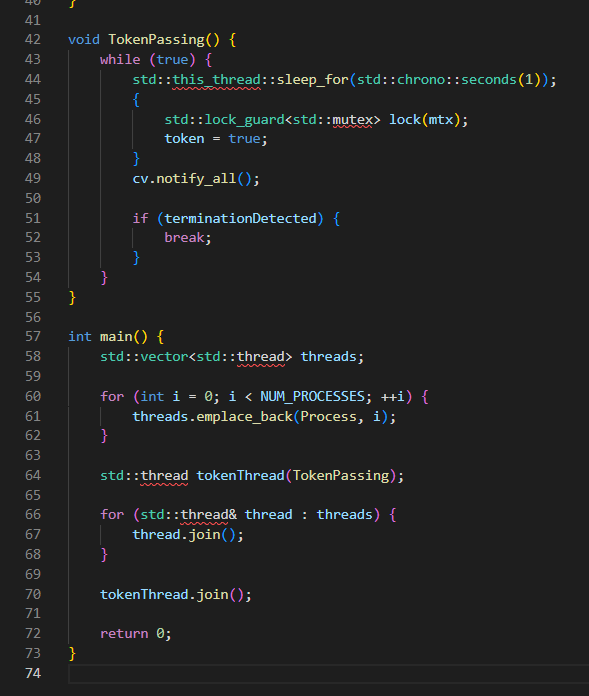


**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**8-Program to implement termination detection**

Termination detection in a distributed system is a complex topic, and the implementation can vary significantly depending on the context and requirements of the system. However, I can provide you with a simplified example in C++ that demonstrates a basic mechanism for termination detection in a distributed system using the "token-passing" approach. This example represents a simplified distributed system with processes communicating through a token.





**OUTPUT:**



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**9-To implement CORBA mechanism by using C++ program at one end and Java Program on the other.**

Implementing the Common Object Request Broker Architecture (CORBA) involves creating a bridge between different programming languages (C++ and Java in this case) through middleware. CORBA uses an Object Request Broker (ORB) to facilitate communication between objects written in different languages.

